C++程序设计第 8 次作业　大数据 2101 班　李嘉鹏　U202115652　2022 年 11 月 23 日

1. 如果有 3 个异常处理 hi 过程：catch(…)、catch(const void *)、catch(int *)，应如何摆放它们的位置？catch(int *)可以放在 catch(const void *)后面而不影响其捕获异常吗 ？

   catch(int *)，catch(const void *)，catch(…)；不行，因为 const void*类型包含 int*类型

2. int x，显式转换(int)x 的结果是左值还是右值？

   结果是右值。

3. 对于全局变量 const int x = 0，能够使用 *const_cast<int *>(&x) = 3 修改 x 的值吗？用 *(int *)(&x) = 3 可以修改 x 的值吗？

   都不可以，因为 x 是全局变量，即使编译时去除了源类型中的 const，运行时也会出现页面保护访问机制冲突。

4. 说明 static_cast 和 const_cast 的区别。

   static_cast 是静态转换，不能去除源类型的 const 和 volatile 属性；const_cast 是只读转换，可以在源类型中去除或添加 const 和 volatile 属性。

5. 对于 Lambda 表达式：auto f = [x] (int y) -> int { return ++x + y; }，解释 [x] (int y) 中的 x 和 y 的意义，并说明为什么 f 名义上是一个函数，但实际上是一个对象。

   x 是捕获列表中的参数，代表程序在调用 f 时，从当前进程中捕获变量 x，并在匿名类中创建一个同名的实例成员变量 x，对 x 的修改不会修改 x 本身；y 是 f 的参数，实际上是匿名类的重载函数 int operator ()(int y)。对于 f，由于其捕获列表非空，因此其对象名不可能是函数指针类型，不能传给函数指针类型的形参，因此看做一个对象。

6. 类模板的实例化有哪几种方式？

   显式（如：template class A <5>，5 是 int，创建一个 int 类）和隐式（MATRIX <int> a(5)）。

7. 分析下面的程序，指出变量 g1 ~ g4、a1 ~ a6 中，哪些变量的地址是相同的（指向同一个对象）？执行完指令“float &a5 = f<float>();”和“float a6 = f<float>();”后，各变量的值是多少？

```
template<typename T, int x=0>
T g = T(10 + x);
template float g<float>;
template<typename T>
T &f( ) {
      T &a = g<T, 0>;
      return ++a;
}
```

```
float g1 = g<float>;
float &g2 = g<float>;
const float &g3 = g<float>;
const float &g4 = g<float, 4>;
int main()
{
    float a1 = g<float>;
    float &a2 = g<float>;
    const float &a3 = g<float>;
    const float &a4 = g<float, sizeof(float)>;
    float &a5 = f<float>();
    float a6 = f<float>();
}
```

g2，g3，a2，a3，a5 的地址相同，指向的对象相同（10.0f）。g4、a4 的地址相同，指向的对象相同（14.0f）。 g1、a1、a6 分别指向不同的存储单元。

执行 float &a5 = f()后，a1=g1=10，g4=a4=14，g2=g3=a2=a3=a5=11

执行 float a6 = f()后，a1=g1=10，g4=a4=14，g2=g3=a2=a3=a5=a6=12

8. 设计一维数组模板 ARRAY，可以实例化各种简单类型的一维数组。在模板的成员函数中需要考虑抛出异常（如内存不足、数组下标越界等），并在 mian()中进行简单的测试（包括捕获异常处理）。

```
class Index {
    int index;
public:
    Index(int i) { index = i; }
    int getIndex() const { return index; }
};
struct SHORTAGE : Index {
    SHORTAGE(int i) : Index(i) { }
    using Index::getIndex;
};

template<class T>
class ARRAY {
    T* data;
    int size;
public:
    ARRAY(int n);
    ~ARRAY();
    T& operator[](int);
};
```

```
template<class T>
ARRAY<T>::ARRAY(int n) {
    if (!(data = new int[size = n]))
        throw SHORTAGE(0); //抛出内存不足异常
}
template<class T>
ARRAY<T>::~ARRAY() {
    if (data) delete[] data;
    data = nullptr;
    size = 0;
}
template<class T>
T& ARRAY<T>::operator[](int i) {
    if (i < 0 || i >= size) throw Index(i); //抛出超出数组下界异常;
    return data[i];
}

int main() {
    ARRAY<int> a(100);  //实例化
    try
        a[200] = 30;
    catch (SHORTAGE)
        cout << "SHORTAGE: Shortage of memory!";
    catch (const Index r)
        cout << "INDEX: Bad index is" << r.getIndex();
    catch (...)
        cout << "ANY: any error caught!";
    return 0;
}
```

9. 设计二维数组模板 ARRAY2，可以实例化各种简单类型的二维数组。模板中包含一个非类型形参，表示构造函数的一个参数的缺省值，用于初始化数组元素的值。并给出任意一条实例化模板的语句。

```
template<class T, int num = 0>
class ARRAY2 {
    T** data; //创建二维数组
    int col;
    int row;
```

```
public:
    ARRAY2(int m, int n);
    ~ARRAY2();
    T& getData(int m, int n);
};

template<class T, int num>
ARRAY2<T, num>::ARRAY2(int m, int n) {
    data = new T * [row = m];
    for (int i = 0; i < m; i++)
        data[i] = new int[col = n];
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            data[i][j] = num;
}
template<class T, int num>
T& ARRAY2<T, num>::getData(int m, int n) {
    if (m < 0 || m >= row || n < 0 || n >= col)
        throw "Illegal!";
    else
        return data[m][n];
}
template <class T, int num>
ARRAY2<T, num>::~ARRAY2() {
    if (data)
        delete[] data;
    data = nullptr;
    col = 0;
    row = 0;
}

int main() {
    ARRAY2<int, 3> f(2, 3); //实例化
    return 0;
}
```